# 8x931Ax, 8x931Hx Universal Serial Bus Adapter Board User's Guide

**December 1997**

**Order Number: 273136-002**

## Figures



## Tables

## 1.0 INTRODUCTION

This guide describes how to connect the 8*x*931A*x*, 8*x*931H*x* Universal Serial Bus Adapter Board (USB931AxHxADBD) to the 8x93x Family USB Evaluation Board. The USB931AxHxADBD Adapter Board comes equipped with the 8*x*931H*x* peripheral controller but supports both the 8*x*931A*x* and 8*x*931H*x* peripheral controllers. The 8*x*931A*x* peripheral controller acts only as a function, whereas the 8*x*931H*x* acts both as a hub and a function. The 8*x*931H*x* supports one upstream USB port and four external downstream USB ports.

**NOTE**

The USB931AxHxADBD adapter board is designed **only** to be used with the 8*x*931A*x* and 8*x*931H*x* peripheral controllers. Due to pin incompatibilities, earlier versions of Intel's USB chips cannot be used with the USB931AxHxADBD Adapter Board.

**WARNING**

The USB931AxHxADBD Adapter Board can only be used with the four-port 8x93x Family USB Evaluation Board Family USB Evaluation Board.

## 2.0 INSTALLATIONS

### 2.1 Installing the CPLD

Replace the installed CPLD ( socket U8 of the 8x93x Family USB Evaluation Board) with the CPLD that was shipped with the USB931AxHxADBD kit. This upgraded CPLD has the same pinout as the original CPLD, however, it contains updated code to correctly set up the memory map of the 8x931 peripheral controller to Harvard Architecture. For more information, see Section 4.0, Memory Maps (pg. 13).

### 2.2 Installing the Adapter Board

1. Remove the 8x930Ax device installed on the 8x93x Family USB Evaluation Board from the U3 socket prior to installing the adapter board. Refer to Figure 1 for the location of socket U3.
2. Place the adapter board on the evaluation board, ensuring that the connectors are properly aligned, and press the adapter board onto the evaluation board until the connectors are fully engaged. Refer to Figure 2 for the correct orientation of the adapter board.

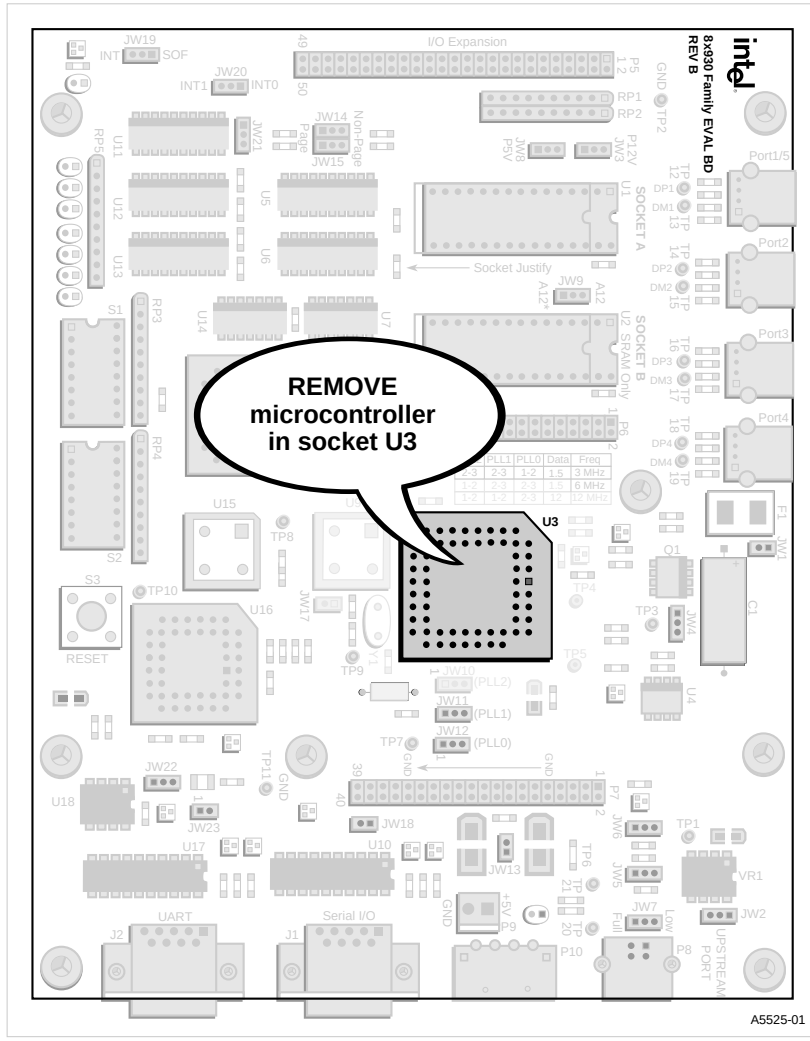



**Figure 1. 8x930Ax Peripheral Controller Location**

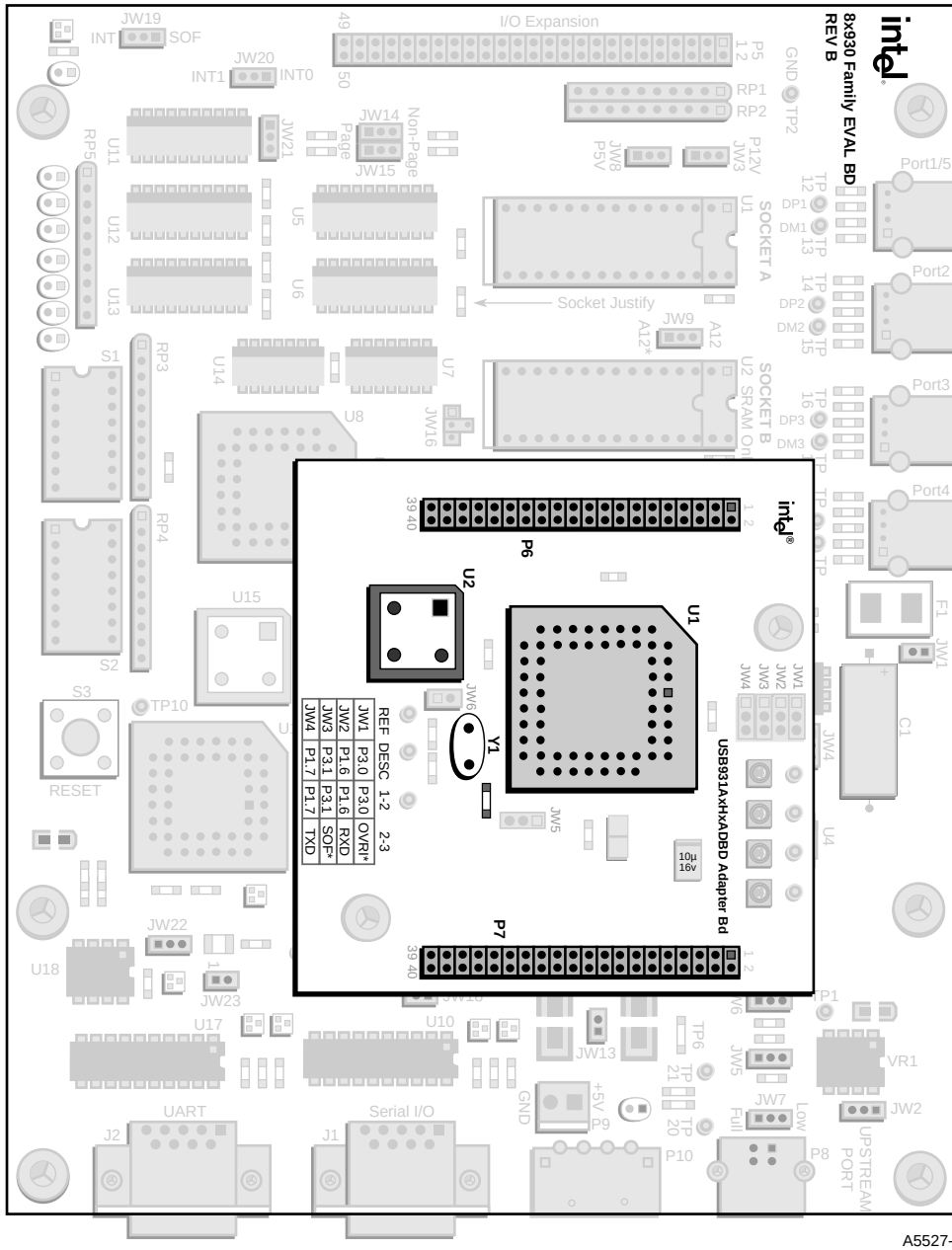



**Figure 2. Adapter Board Orientation**

## 2.3 Installing the EPROM

You need to replace the EPROM that is installed in Socket A of the 8x93x Family USB Evaluation Board with the EPROM that was shipped with the USB931AxHxADBD kit. This upgraded EPROM has the same pinout as the original, however, it contains updated RISM (Reduced Instruction Set Monitor) code and firmware to support the 8x931 family of peripheral controllers.

**NOTE**

Socket A can accept either a 32K, 64K, or 128K EPROM. Because of this, when you install the 32K EPROM provided by Intel, there will be four holes left open by the notch. See Figure 3 for the correct orientation of the EPROM.

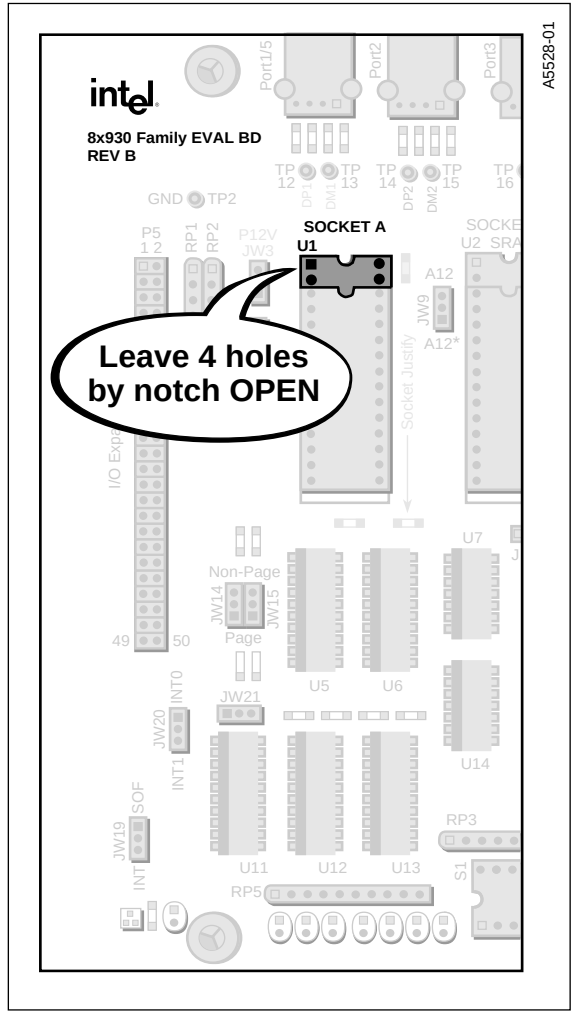



**Figure 3. EPROM Orientation**

## 2.4 Installing the 8*x*931A*x* Peripheral Controller

1. Remove the 8*x*931H*x* device installed on the USB931AxHxADBD adapter board from the U1 socket. Refer to Figure 4 for the location of socket U1.
2. Install the 8*x*931A*x* in socket U1 insuring proper alignment.

## 2.5 Mandatory Jumper and Dip Switch Setting Changes

The following two tables list the mandatory jumper and dip switch settings that must be set on the 8x93x Family USB Evaluation Board in order for the 8*x*931A*x*, 8*x*931H*x* Adapter Board to function properly. Section 3.5 describes the jumper settings of the 8*x*931A*x*, 8*x*931H*x* Adapter Board.

**Table 1. Mandatory Jumper Settings on the 8x93x Family USB Evaluation Board**

| DIP Switch | Position | Description |
|---|---|---|
| JW8 | 2-3 | Vcc to socket A pin 30 |
| JW14 | 1-2 | Nonpage mode |
| JW15 | 1-2 | Nonpage mode |

**Table 2. Mandatory Dip Switch Settings on the 8x93x Family USB Evaluation Board**

| DIP Switch | Position | Description |
|---|---|---|
| RD1 | OFF | 64K 8051 compatible |
| RD0 | OFF | 64K 8051 compatible |
| ADS2 | ON | 32K Byte EPROM |
| ADS1 | ON | 32K Byte EPROM |
| ADS0 | OFF | 32K Byte EPROM |
| ADT1 | ON | EPROM in socket A |
| ADT0 | ON | EPROM in socket A |
| EA# | ON | External ROM |
| MOD1 | ON | Binary nonpage mode |
| MOD0 | OFF | Binary nonpage mode |
| RISM | ON | RISM is on |

**NOTE**

For more details on each setting, refer to the *8x93x Family USB Evaluation Board User's Manual (order number 27297202)* that was shipped with the base board.

**WARNING**

Since the 8x93x Family USB Evaluation Board supports all USB families, the default jumper and dip switch settings on the shipped base board may be incorrect for use with this 8x931 family adapter board. You must first verify that the above settings match the base board for correct operation of the adapter board.

### 2.5.1 Power On Test

With the serial cable disconnected, applying power by pressing the reset button causes RISM to initialize the evaluation board, ending with a sequence display on the LEDs. The LEDs will first light up on the 8x93x Family USB Evaluation Board followed by the 8*x*931A*x*, 8*x*931H*x* Adapter Board. All the LEDs will light up then turn off following the sequence described in Table 3 below:

**Table 3. Beginning-to-End LED Sequence**

| Board | LED Sequence |
|---|---|
| 8x93x Family USB Evaluation Board | P1.5, P.1.4, P1.3, P1.2, P1.1, P1.0 |
| 8x931Ax, 8x931Hx Adapter Board | LED3, LED2, LED1, LED0 |

Once the LEDs have flashed on and off, they should end in the following state:

**Table 4. Ending State of LEDs**

| Board | LED's Left On | LED's Left Off |
|---|---|---|
| 8x93x Family USB Evaluation Board | P1.4 P1.2 P1.0 | P1.5 P1.3 P1.1 |
| 8x931Ax, 8x931Hx Adapter Board | | LED3 LED2 LED1 LED0 |

**NOTE:** P1.6 and P1.7 may remain on after the Power On test. Ignore P1.6 and P1.7 status during the Power On sequence.

## 3.0 BOARD LAYOUT

Figure 4 shows the layout of the board and its major components.

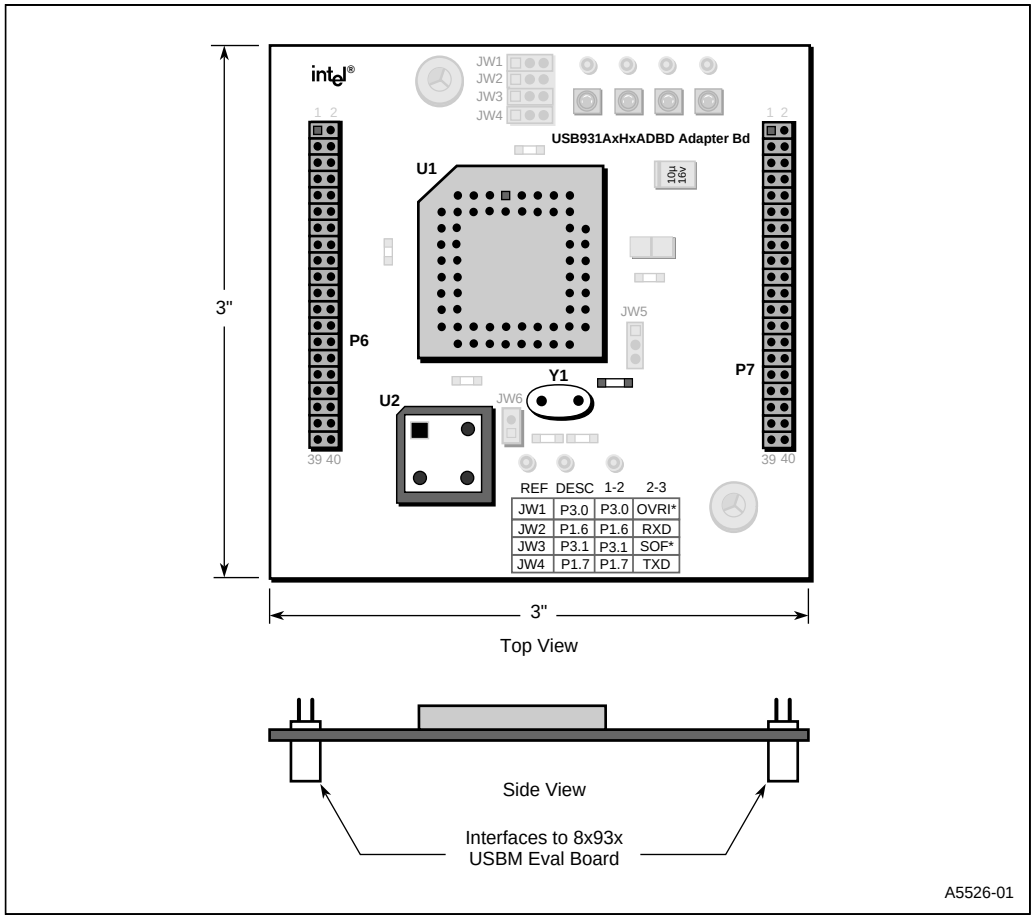



**Figure 4. Adapter Board Layout**

**Table 5. Part Association**

| PART | DESCRIPTION |
|---|---|
| U1 | 8x931xx peripheral Controller |
| U2 | 12.000 Mhz crystal |
| JW1 | P3.0 / OVRI# |
| JW2 | P1.6 / RxD |
| JW3 | P3.1 / SOF# |
| JW4 | P1.7 / TxD |
| JW5 | PLLSEL |
| JW6 | Clock Select |
| TP1 / CR1 | LED 3 |
| TP2 / CR2 | LED 2 |
| TP3 / CR3 | LED 1 |
| TP4 / CR4 | LED 0 |
| TP5 | CLK 12MHz |
| TP6 | XTAL 2 |
| TP7 | AVCC |
| P6 | Header |
| P7 | Header |

Headers P6 and P7are available as test points for various signals and user expandability. Refer to Section 6.0, 8x931xx ADAPTER BOARD SCHEMATICS (pg. 23) for the exact pinout of each header.

## 3.1 FEATURES

## 3.2 CPU Foot Print

The 8*x*931A*x*, 8*x*931H*x* Adapter Board has the foot print for a 68-pin PLCC socket to match the pinouts of both the 8*x*931A*x* and the 8*x*931H*x* peripheral controllers.

## 3.3 External Clock Oscillator/Crystal

The board provides space for a crystal and a capacitor to use the internal oscillator of the 8*x*931A*x* or the 8*x*931H*x* and also contains a clock oscillator that produces an independent external clock source for either the 8*x*931A*x* or the 8*x*931H*x*. Only one clock source can be used at a time. By default, the clock oscillator is used and the crystal and capacitor are not populated. To use the internal oscillator, you must cut jumper JW6 and install the appropriate crystal and capacitor.

## 3.4 Test Points

The adapter board contains test points for: CLK12MHZ, XTAL2, AVCC, LED0, LED1, LED2, and LED3. (LED[3:0] are for the keyboard. In order to use them, you must enable the KSEN bit in the KBCON register.)

## 3.5 Jumpers

The board contains jumpers to allow both the 8*x*931A*x* and 8*x*931H*x* devices to be reconfigured. Four 3-pin jumpers are used to set up the features of the adapter board. Table 6 below lists the jumper configurations.

**Table 6. Jumper Configurations**

| Jumper | Position (1-2) | Position (2-3) |
|---|---|---|
| JW1 | P3.0 | OVRI# |
| JW2 | P1.6 | RxD |
| JW3 | P3.1 | SOF# |
| JW4 | P1.7 | TXD |

**NOTE**

Due to the pinout of the 8*x*931A*x* and the 8*x*931H*x*, the information on four pins had to be redirected from the mapping of the pinout of the 8x930Ax. On the 8x930 peripheral controller, the serial port was multiplexed with Port 3, pins 0 and 1; but on the 8x931 peripheral controller it is multiplexed with Port 1 pins 6 and 7. This leads to the following restricted jumper configurations:

- JW1 in position 1-2 with JW3 in position 2-3.
- JW2 in position 1-2 with JW4 in position 2-3.

## 3.6 Evaluation Board Interface

The adapter board interfaces to the 8x93x Family USB Evaluation Board through two 40-pin female headers. The headers are surface mounted on the solder side of the board. The interface is "duplicated" physically and electrically on the component side of the board to allow probing access.

## 3.7 Serial Communications

The adapter board supports both the 8*x*931A*x* or 8*x*931H*x* USB chip. These devices have a default start-up core frequency of 3 MHz. You can change this frequency to 6 MHz by clearing the LC bit located in the PCON register. You can switch back to 3 MHz by setting the LC bit. Please refer to the *8x931AA, 8x931HA Universal Serial Bus Peripheral Controller User's Manual* (order number 273102-01) for further information.

The flexibility of frequency changes for power saving has a direct impact on baud rate and serial communication. To allow debugging sessions without risk of board-PC communication loss, a number of changes have been added to the RISM code. To establish communication between the evaluation board and the debugger (Keil, PLC, or Tasking software), use either the external serial link or the internal serial link.

### 3.7.1 Using the External Serial Port

**It is strongly recommended to use the external-serial port to communicate with the debugger environment**. When you use the external serial port, the baud rate is 9600 (make sure you set the debugger software to use the same rate). With all internal timers free, you can use them for your specific application. The internal serial port is then free and can be used by your application. One interrupt (INT0), however, is used by the UART and cannot be used. The priority should **not** be changed.

The external serial port is labeled "UART" on the board. To enable this port, set the UARTC switch to the "ON" position. When using this port, programmers can set or clear the LC (low clock) bit in their code. Since the PCON register is not bit addressable, programmers cannot use the SETB or CLR instructions to change the LC bit.

The programmer should use one of the following:

- "ORL PCON, #20h" or "LCALL 0083" to set the LC bit.
- "ANL PCON, #DFh" or "LCALL 008B" to clear the LC bit.

### 3.7.2 Using the Internal Serial Port

**It is NOT recommended to use the internal serial port to communicate with the debugger**. However, if you use this serial port, the baud rate is fixed at 9600 (set the debugger software to use the same rate). RISM uses timer-two to generate the baud rate, making this timer unavailable for your application. The internal serial port, RxD and TxD, is multiplexed with Port 1, bits 6 and 7, making those bits unavailable.

Both external interrupts, INT0 and INT1, are free during this time. When using the internal serial port, the programmer can set or clear the LC bit, however, in doing so, the programmer cannot use the SETB and CLR instructions. You must use a long call to a routine inside the RISM that sets or clears the LC bit.

The programmer should use the following:

- "LCALL 0083" to set the LC bit.
- "LCALL 008B" to clear the LC bit.

## 3.8 Keyboard

In order for the 4 LEDs on the adapter board to function, you must set the KSEN bit in the KBCON register. Please refer to the *8x931Ax, 8x931Hx Universal Serial Bus Peripheral Controller User's Manual* for more information.

## 3.9 Firmware

This section describes the details on the firmware shipped on the EPROM. This EPROM contains two major sections of code:

1.) USB931RISM (Section 5.0, RISM (pg. 14)).

2.) USB931FIRMWARE (Section 3.9.1, USB931FIRMWARE (pg. 12)).

### 3.9.1 USB931FIRMWARE

This code provides the user with working embedded function and hub firmware. Details on the code structure can be obtained by downloading the USB931 Firmware at http://www.intel.com/design/usb/swsup/

### 3.9.2 USB931Firmware Operating Instructions

In Section 2.5, you set the Evaluation Board up to use RISM. In this section you will be shown how to set the board up to be USB931FIRMWARE. Located on S2 on the 8x930 Family Evaluation Board, set the MOD1 dip switch to OFF.

This action will read the memory map on the EPROM to point to the HUB code. RISM starts from :0000H on the EPROM. USB931Firmware code starts from :6000H on the EPROM. This firmware runs both embedded function and HUB functionality. The LED sequence will remain off. Testing of this firmware requires a PC with USB-ready software.

The default setting for the USB931Firmware is embedded function. For instructions on use of the HUB function see Section 3.9.3.

### 3.9.3 HUB Function

For HUB function, using the I/O Expansion header on 8x930 Family Evaluation Board, place a jumper from GND to P3.3.

### 3.9.4 Running 8x931Ax device in Low-Speed USB Data Rate

On the USB931AxHxADBD Adapter Board, place a jumper between pin 13 and pin 14 on the P7 header (on the right edge of the adapter board).

# 4.0 MEMORY MAPS

## 4.1 Internal Memory Map

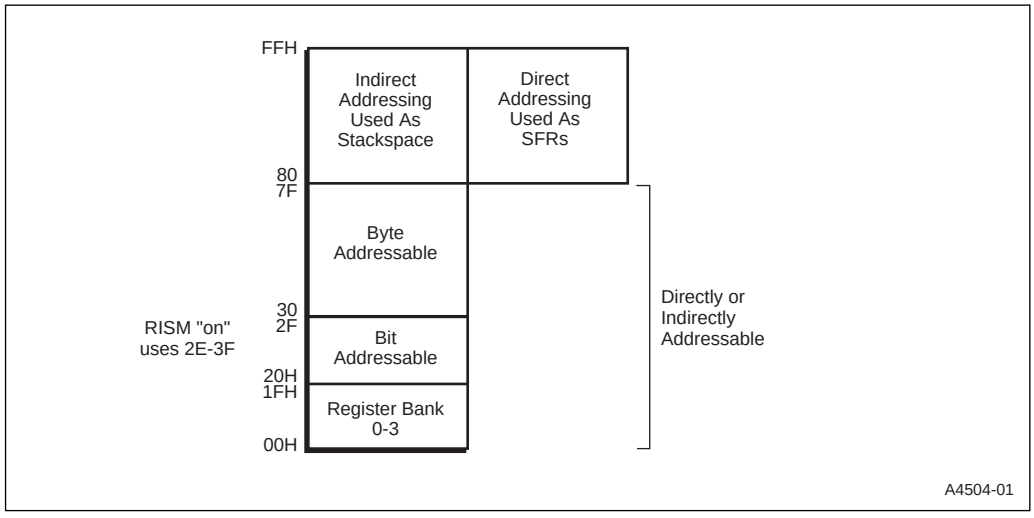



**Figure 5. Internal Memory Map RISM "On"**

## 4.2 External Memory Map with RISM "On"

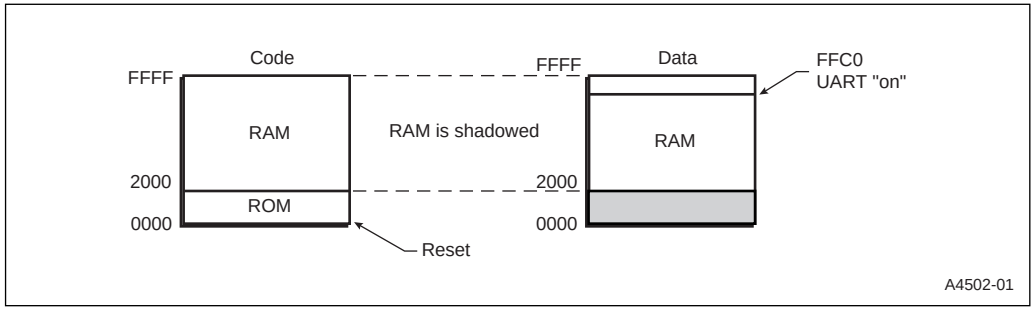
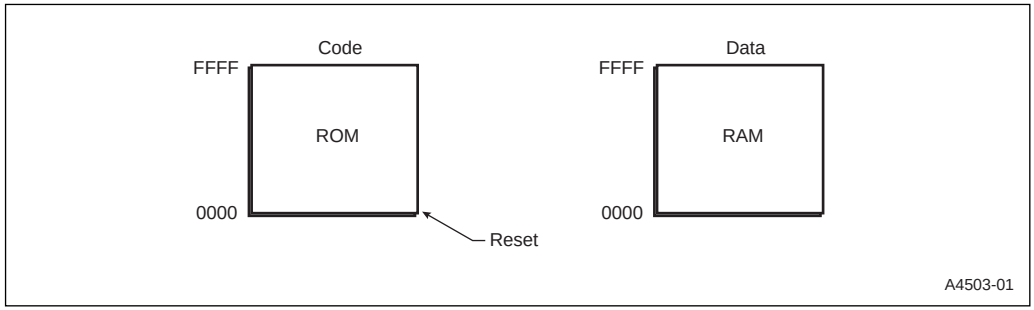



**Figure 6. External Memory Map RISM "On"**

## 4.3 External Memory Map with RISM "Off"




## 4.4 External Memory Map RISM "Off" Explanation of the Memory Maps

Since the 8*x*931A*x* and 8*x*931H*x* are built from a MCS® 51 core, the memory structure is of Harvard Architecture. By definition, Harvard Architecture has separate code and data spaces. The code space can only be read by using the program counter, whereas data spaces are read from and written to using the data pointer (DPTR) for external data. For this reason we have a PLD on the evaluation board. With the RISM 'on', a portion of the RAM is mapped into code space (2000h to FFFH). This allows the downloading of user code as data by using the DPTR, and execution of the code through the program counter. Since code and data spaces are overlapped when RISM is 'on', verify that external memory is **not** using the same data spaces that code is using. Additionally, the new PLD provides for the traditional Harvard Architecture when the RISM dip switch is in the 'off' position.

## 5.0 RISM

The RISM (Reduced Instruction Set Monitor) consists of a program located in the internal ROM of the peripheral controller or other ROM on an evaluation board. This program provides simple operations such as: read memory, write memory, start execution, and stop execution. The communication software running on the host computer uses RISM commands to provide a complete user interface to the target board. It is part of a debugger environment.

The ROM-monitor communicates with the peripheral controller by sending special RISM commands across the serial port from a personal computer (PC) at a fixed baud rate. Figure 7 shows the relationship between the evaluation board and a PC.

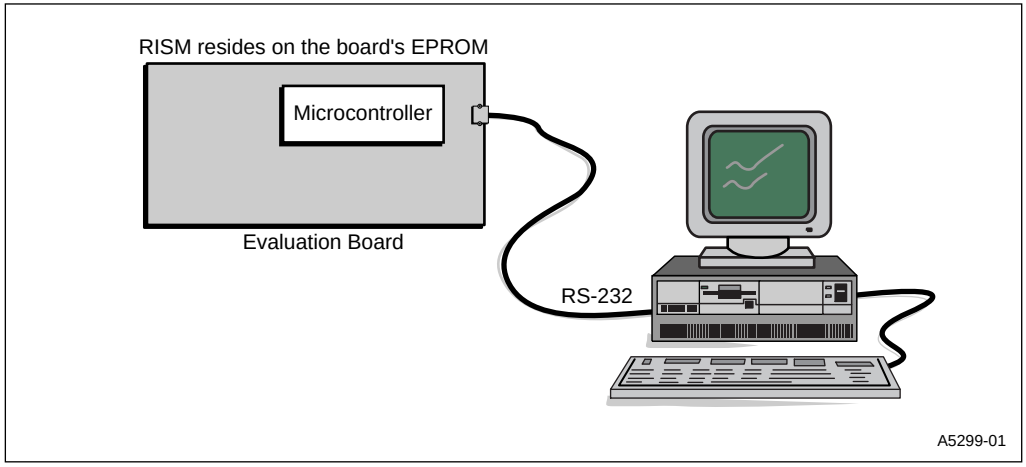



**Figure 7. Serial Link Between the 8x931 Family USB Evaluation Board and a PC**

**NOTE**

The structure and functionality of RISM described in this document is related to the MCS® 51 peripheral controllers and 8x931 Family USB devices. A similar implementation is used for the MCS® 251 peripheral controllers and 8x930 Family USB devices. For more specifics, refer to the *8x931AA, 8x931HA Universal Serial Bus Peripheral Controller User's Manual*.

## 5.1 RISM Structure and Functionality

Upon reset, RISM runs an initialization section of code. Then, it waits for commands from the host PC. The peripheral controller enters a waiting loop called "Monitor_Pause" in which it waits to receive data characters across the serial port. A received character can either be a data or RISM command. Each command is one byte in length and has a value between 00H and 1FH. If you send data values less than 1FH, you **must** precede them with a Data Latch Enable (SET_DLE) command to alert RISM that the next character should be stored in the DATA register.

When a receive interrupt occurs, RISM checks to see if the DLE flag is set or if the data received is greater than 1FH. If either of these conditions are true, then RISM assumes that the received byte is data, reading it into a 32-bit first-in-last-out (FILO). Each time new data comes in, the DATA register shifts left by eight bits. If RISM identifies the received data as a command, RISM executes it. After completing the command, the device re-enters the "Monitor_Pause" loop. Figure 8 shows the overall flow of the RISM code.

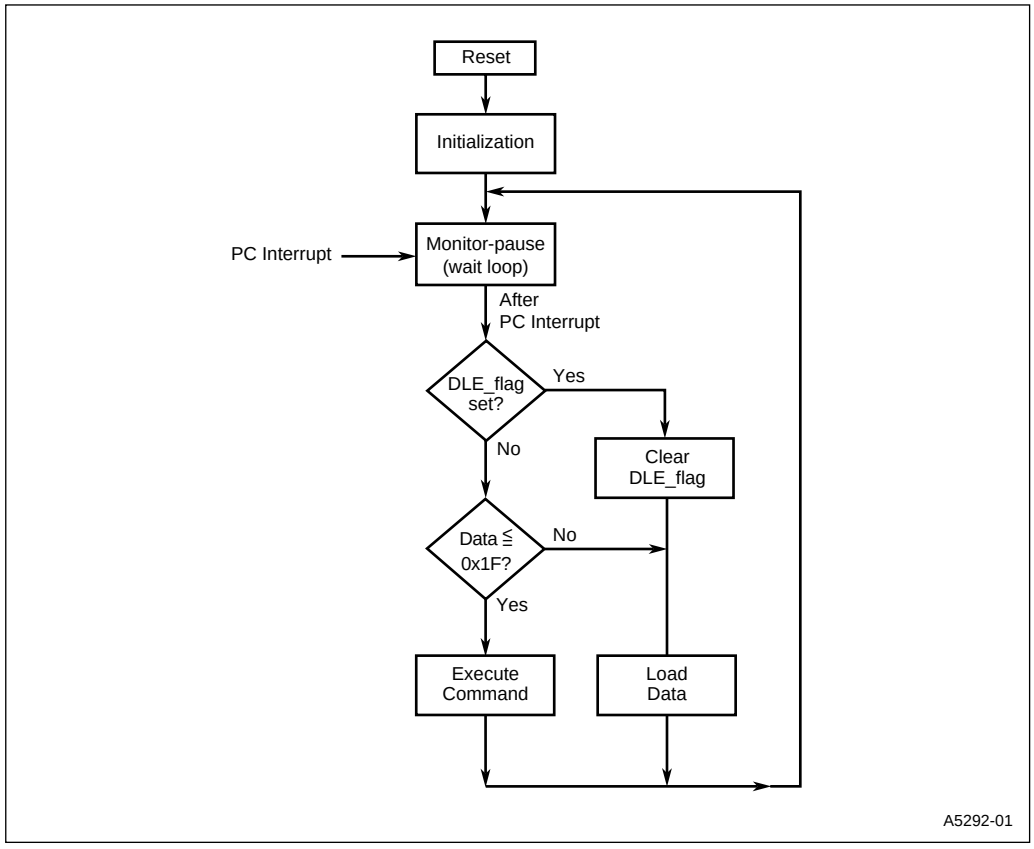



**Figure 8. Flow of RISM Code**

## 5.2 RISM Resources

**WARNING**

Because the evaluation board and RISM use some of the same resources to communicate with the host PC, these resources will not be available to you. The restrictions on the availability of these resources depends on the board's application. Some restrictions may not apply if you are using the board in stand-alone mode.

The affected resources are:

- RISM reserves RAM locations 2Eh–39h for RISM data registers. If the application program attempts to use these locations, RISM will not run properly.
- Memory locations 80h to 8Fh are reserved for RISM stack.
- RISM can handle communication to the debugger through the external UART or the internal SIO.
  - — The external UART uses either interrupt INT0 or INT1. The jumper settings on the evaluation board determine which interrupt RISM uses. The interrupt that RISM uses becomes unavailable for application programs. The USB evaluation board's default interrupt setting is INT0.
  - — If the internal SIO is used, then one of the timers (1 OR 2) also becomes unavailable to the user. The USB evaluation board's default timer setting is 2. Port 1, bits 6 and 7, are also unavailable.
- RISM has a release number byte at code memory location 00FFh.
- Code memory 0000h - 1FFFh.

## 5.3 General RISM Information

RISM resides in the first 4Kbytes of the 32-Kbyte EPROM. In order for the adapter board to function, the jumpers must be set as stated in Section 2.5, Mandatory Jumper and Dip Switch Setting Changes (pg. 5). This sets the evaluation board to operate in binary, nonpage mode. The mode DIP-switch settings **must** correspond to the memory map.

The host PC debugger sends its commands to RISM, and the 8x93x Family USB Evaluation Board executes them. RISM downloads user object code, one byte at a time, into the SRAM; starting at location 4000h. RISM also sets the user's program counter to the beginning of the user code when it receives either a GO or RUN command from the debugger.

RISM source code is available to the public at no cost and with no technical support. The code is available via the following World Wide Web site: *http://www.intel.com/design/usb/swsup.*

## 5.4 RISM Stack Pointer Area

RISM sets the stack pointer to location 80h. You can initialize the stack pointer to a different area; however, you should leave approximately 16 bytes for RISM to use for debugging. RISM saves and restores the user stack during the execution of debugger commands.

## 5.5 RISM Configuration

**NOTE**

This section applies to the reset of RISM **only**.

The RISM code in this USB evaluation board has been configured to meet the 8x931 USB peripheral controller architecture, but can also be reconfigured for MCS® 251 boards. The code is available via the following World Wide Web site: *http://www.intel.com/design/usb/swsup***.**

The board ships with the following RISM-code configuration:

- **EXT_MEMORY**: Configured as 4000H.
- **EXTINTR**: Configures the external UART so it can use INT0 (equ 0).
- **TIMER**: Configures the baud rate generator for timer 2 (equ 2).

## 5.6 RISM Registers and Bits

RISM defines (in internal RAM locations) registers that are used throughout the code to allow navigation through the RISM structure, data storage, and context switching.

The registers are:

- mode
- selector
- RISM_DATA
- RISM_ADDR
- user_pc
- Baud_rate

The registers **mode** and **selector** are used in the form of bits. Each bit serves a special purpose in running the RISM code and communicating with it. Review the RISM code to learn more about these registers and their bits.

## 5.7 Interrupt Enable and Priority Registers

**WARNING**

Do not clear the global enable bit or all of the maskable interrupts will not function. If you are using the internal serial port, do not clear the serial-port interrupt enable bit. If you are using the external UART, do not clear int0 interrupt enable bit. Clearing any or all of the bits will terminate communication between the evaluation board and the host PC.

Change the contents of the interrupt enable registers by using logical OR or AND operations. For more details about interrupts, refer to the *8x931AA, 8x931HA Universal Serial Bus Peripheral Controller User's Manual*.

## 5.8 Interrupt Vectors Offset

All available application interrupt vectors are redirected to:

EXT_MEMORY + vector_offset

(Where *EXT_MEMORY* = 4000H on the evaluation board.)

(Where *vector_offset* is the real ISR Vector Address.)

## 5.9 RISM Commands

RISM has 32 possible commands; each command has an associated code. The code is a hexadecimal number between 00 and 1F inclusive. In addition, each command has a special function represented by a set of instructions in RISM. Table 7 shows the set of RISM commands and explains each command's task(s).

**Table 7. RISM Commands**

| Value | Name | Code | Description |
|---|---|---|---|
| 1 | Set_DLE | 00H | Sets the DLE flag. This forces the next byte received by the RISM to be treated as data even if its value corresponds to a RISM command. The DLE flag is cleared as soon as the byte is read. |
| 2 | Xmit | 01H | Transmits the lower-eight bits of the 32-bit DATA register to the host, and shifts the DATA register eight places to the right. |
| 3 | Xmita | 02H | Transmits the lower-eight bits of the 32-bit DATA register to the host, shifts the DATA register eight places to the right, and increments the ADDR register by one. |
| 4 | RESERVED | 03H | Reserved |
| 5 | Read_byte | 04H | Reads the byte of memory pointed to by the ADDR register and places the result in the least significant byte of the DATA register. |
| 6 | Read_word | 05H | Reads the word of memory pointed to by the ADDR register and places the result in the least significant word of the DATA register. SFR's **cannot** be read from as words. |
| 7 | Read_long | 06H | Reads the double-word of memory pointed to by the address register and places the result in the 32-bit DATA register. SFRs **cannot** be read from as words. |
| 8 | Write_byte | 07H | Stores the least significant byte of the DATA register in the byte of memory pointed to by the ADDR register and increments the ADDR register by one, so it points at the next memory byte. |

**Table 7. RISM Commands (Continued)**

| Value | Name | Code | Description |
|---|---|---|---|
| 9 | Write_word | 08H | Stores the least significant word of the DATA register in the word of memory pointed to by the ADDR register and increments the ADDR register by two, so it points to the next memory word. SFR's **cannot** be written to as words. |
| 10 | Write_long | 09H | Stores the 32-bit DATA register in the double-word of memory pointed to by the ADDR register and increments the ADDR register by four, so it points at the next memory double-word. SFRs **cannot** be written to as words. |
| 11 | Data_to_addr | 0AH | Loads the 32-bit ADDR register with the 32-bit DATA register.<br>REGISTER and SFR addresses have byte (8-bit) address. Byte registers have addresses 0x00 to 0x0F; and word registers have addresses 0x00 to 0x1E. Address 0x38 (DR56) is the only addressable dword register address now. SFRs have addresses 0x80 to 0xFF. Other memory locations should use 32-bit addresses. |
| 12 | Indirect0 | 0BH | Reads the memory word pointed to by the ADDR and stores it into the ADDR register. |
| 13 | Read_PSW | 0CH | Loads the low word of the 32-bit DATA register with the PSW and PSW1 associated with the user's code. PSW loads at the lowest byte of the DATA register. |
| 14 | Write_PSW | 0DH | Loads the PSW and PSW1 associated with the user's code from the low word of the DATA register. PSW loads at the lowest byte of the DATA register.<br>PSW1 is updated last, so shared bits of PSW and PSW1 are dictated by PSW1. |
| 15 | Read_SP | 0EH | Loads the DATA register with the SP (Stack Pointer) associated with the user's code. |
| 16 | Write_SP | 0FH | Loads the SP (Stack Pointer) associated with the user's code from the DATA register. |
| 17 | Read_PC | 10H | Loads the 32-bit DATA register with the 32-bit user_pc (Program Counter) RISM register associated with the user's code. |
| 18 | Write_PC | 11H | Loads the 32-bit user_pc (Program Counter) RISM register associated with the user's code from the 32-bit DATA register. |
| 19 | GO | 12H | Starts the execution of user code at the address in the user_pc register. As long as the user program does not disable interrupts, the monitor will run in the background servicing serial-port interrupts. |

**Table 7. RISM Commands (Continued)**

| Value | Name | Code | Description |
|---|---|---|---|
| 20 | Halt | 13H | Stops the execution of user code and returns to the RISM "Monitor_Pause" loop. The user Program Counter restores in the user_pc (Program Counter) RISM register. |
| 21 | Report | 14H | Loads the lowest byte of the DATA register with status information and transmits the value back to the host. (Status values are 00=stopped, 01=running, 02=trapped[at break].) |
| 22 | RESERVED | 15H | Reserved |
| 23 | RESERVED | 16H | Reserved |
| 24 | Read_baud | 17H | Reads Baud_rate1 and Baud_rate into the low word of the DATA register.<br>• For external UART, DATA+3 contains the low byte of the Decimal Divisor whereas DATA+2 contains the high byte of the Decimal Divisor.<br>• For TIMER 1 as baud generator, DATA+3 contains the reload value and DATA+2 contains the SMOD1 value (0 or 1).<br>• For TIMER 2, DATA+3 and DATA+2 contain the RCAP2L and RCAP2H value respectively. |
| 25 | Write_selector | 18H | Loads the SELECTOR byte with the lowest byte of the DATA register. The valid selector register values are 01= for REGs, 08=for SFRs, and 00= for other locations. **Execute this command before commands 5, 6, 7, 8, 9, 10.** |
| 26 | RESERVED | 19H | Reserved |
| 27 | RESERVED | 1AH | Reserved |
| 28 | Step | 1BH | Single-steps one instruction of user code and returns to RISM "Monitor_Pause". The user code Program Counter for next instruction is restored in the RISM register user_pc. |
| 29 | Write_baud | 1CH | Writes the low word of the DATA register to Baud_rate and Baud_rate1 and sets the baud rate. It assumes the value of the baud rate is already in the DATA register.<br>• For external UART, DATA+3 contains the low byte of the Decimal Divisor whereas DATA+2 contains the high byte of the Decimal Divisor.<br>• For TIMER 1 as baud generator, DATA+3 contains the reload value and DATA+2 contains the SMOD1 value (0 or 1).<br>• For Timer 2, DATA+3 and DATA+2 contain the RCAP2L and RCAP2H values respectively.<br>(Check READ_BAUD). |

**Table 7. RISM Commands (Continued)**

| Value | Name | Code | Description |
|---|---|---|---|
| 30 | Read_frequency | 1DH | Reads the value of the frequency from SYS_FREQ to the DATA register. The value is read to the DATA register in BCD format representing the frequency in Hz. |
| 31 | Report_device | 1EH | Loads the low byte of the DATA register (data-low) with DEVICE_ID. |
| 32 | RESERVED | 1F | Reserved |

## 5.10 Example of RISM Command Use

**Example:** The goal is to download four bytes of data from the PC to the target board as a long word in the DATA register. The four bytes are: 0x13, 0x22, 0x00, and 0x33.

**Procedure**: Send the following sequence of characters over the serial port.

1. Send 0x00.

   This character corresponds to the Set_DLE command. It tells RISM that the next character should be treated as data.

2. Send 0x13

   This character is less than 0x1F. It is sent after the DLE flag is set. It is treated as data and stored in DATA+3.

3. Send 0x22

   This character is greater than 0x1F. It is stored in DATA+3. The previous character is shifted to DATA+2.

4. Send 0x00

   This character is sent to set the DLE flag so that the next character will be treated as data.

5. Send 0x00

   This character is less than 0x1F. It is sent after the DLE flag is set. It is treated as data and stored in DATA+3. The other data characters are shifted left.

6. Send 0x33

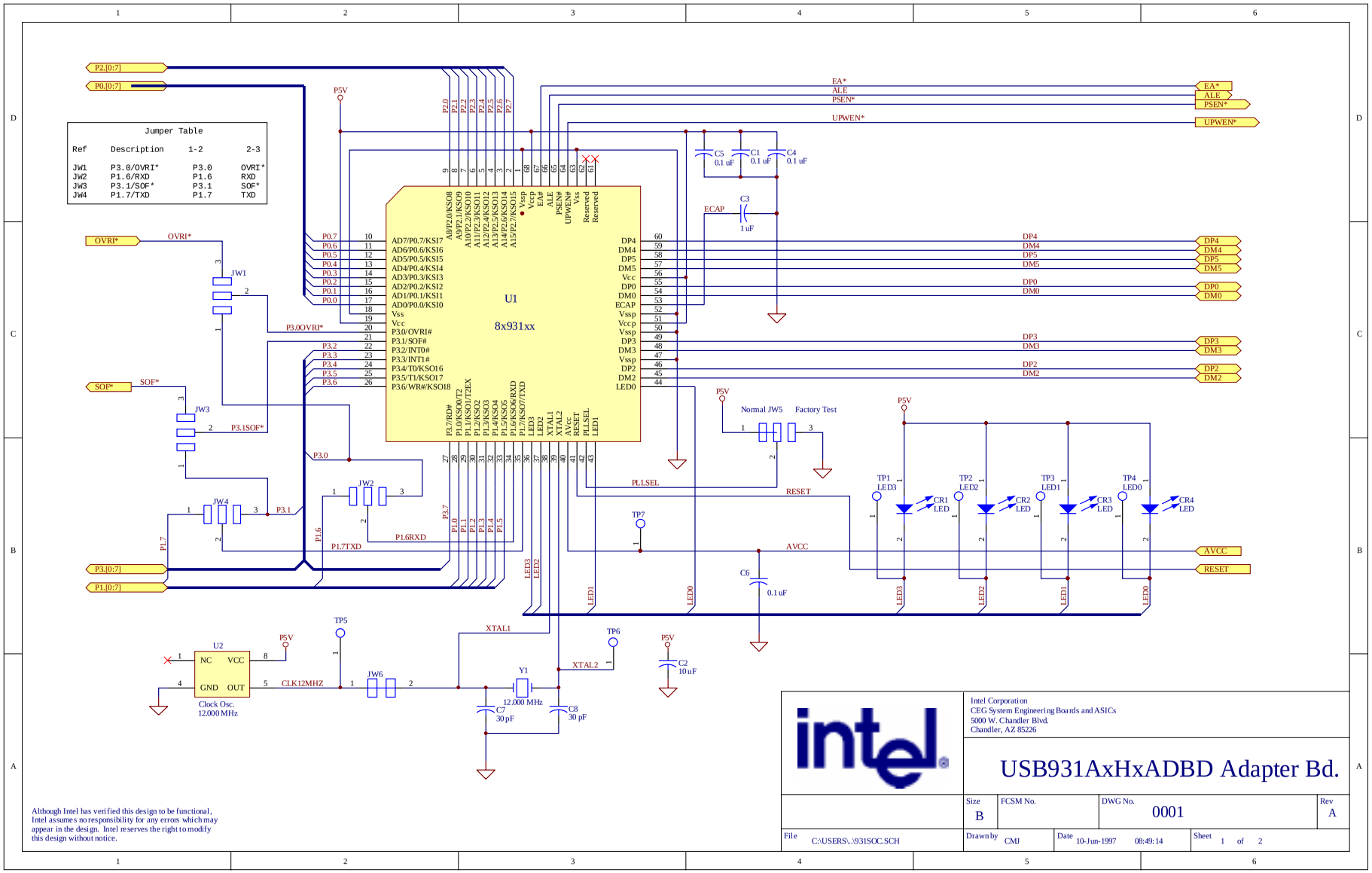
   This character is greater than 0x1F. It is treated as data and stored in DATA+3. The other data characters are shifted left.

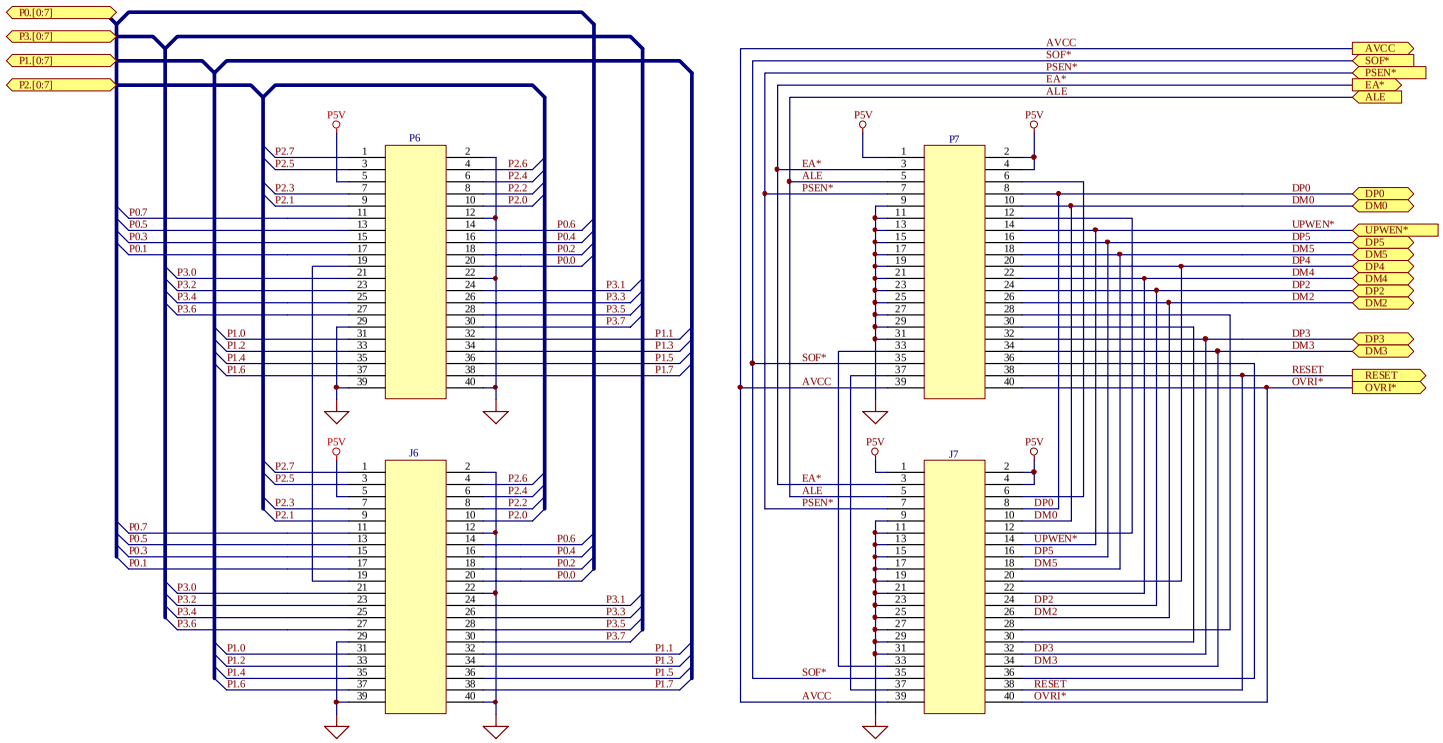
   At this point, we have:

   a. Content (DATA) = 0x13(Highest byte)
   b. Content (DATA+1) = 0x22
   c. Content (DATA+2) = 0x00
   d. Content (DATA+3) = 0x33(Lowest byte)

## 6.0 8X931XX ADAPTER BOARD SCHEMATICS

The pages that follow provide you with schematics for the adapter board. These files are also available on the Intel World Wide Web site at: *http://www.intel.com/design/usb/*.


Jumper Table
Ref Description 1-2 2-3
JW1 P3.0/OVRI* P3.0 OVRI*
JW2 P1.6/RXD P1.6 RXD
JW3 P3.1/SOF* P3.1 SOF*
JW4 P1.7/TXD P1.7 TXD
U1
8x931xx
U2
Clock Osc.
12.000 MHz
Normal JW5 Factory Test
Intel Corporation
CEG System Engineering Boards and ASICs
5000 W. Chandler Blvd.
Chandler, AZ 85226
USB931AxHxADBD Adapter Bd.
Size B
FCSM No.
DWG No. 0001
Rev A
File C:\USERS\..\931SOC.SCH
Drawn by CMJ
Date 10-Jun-1997 08:49:14
Sheet 1 of 2
Although Intel has verified this design to be functional, Intel assumes no responsibility for any errors which may appear in the design. Intel reserves the right to modify this design without notice.




Although Intel has verified this design to be functional, Intel assumes no responsibility for any errors which may appear in the design. Intel reserves the right to modify this design without notice.

Intel Corporation
CEG System Engineering Boards and ASICs
5000 W. Chandler Blvd.
Chandler, AZ 85226

# USB931AxHxADBD Adapter Bd.

| Size | FCSM No. | DWG No. | Rev |
|---|---|---|---|
| B | | 0001 | A |

| File | Drawn by | Date | Sheet |
|---|---|---|---|
| C:\USERS\..\931CON.SCH | CMJ | 10-Jun-1997 08:44:29 | 2 of 2 |